PHP 1° PARTIE :
LES BASES

1. Présentation
2. Historique
3. Fonctionnement
4. Script PHP
5. Exemple
6. Syntaxe
7. Les variables
8. Les variables scalaires
9. Les tableaux
10. Portée d'une variable
11. Les variables statiques et dynamiques
12. Les constantes
13. Les opérateurs et structure de contrôle
14. Affichage

1 . Présentation

A server-side, HTML-embedded scripting language

- Langage de script côté serveur ;
- Embarqué dans les pages HTML;
- Syntaxe héritée du C et du Perl ;
- Extensible (nombreuses bibliothèques et fonctions) ;
- Supporte pratiquement tous les standards du web ;
- Logiciel Open Source.

2 . Historique

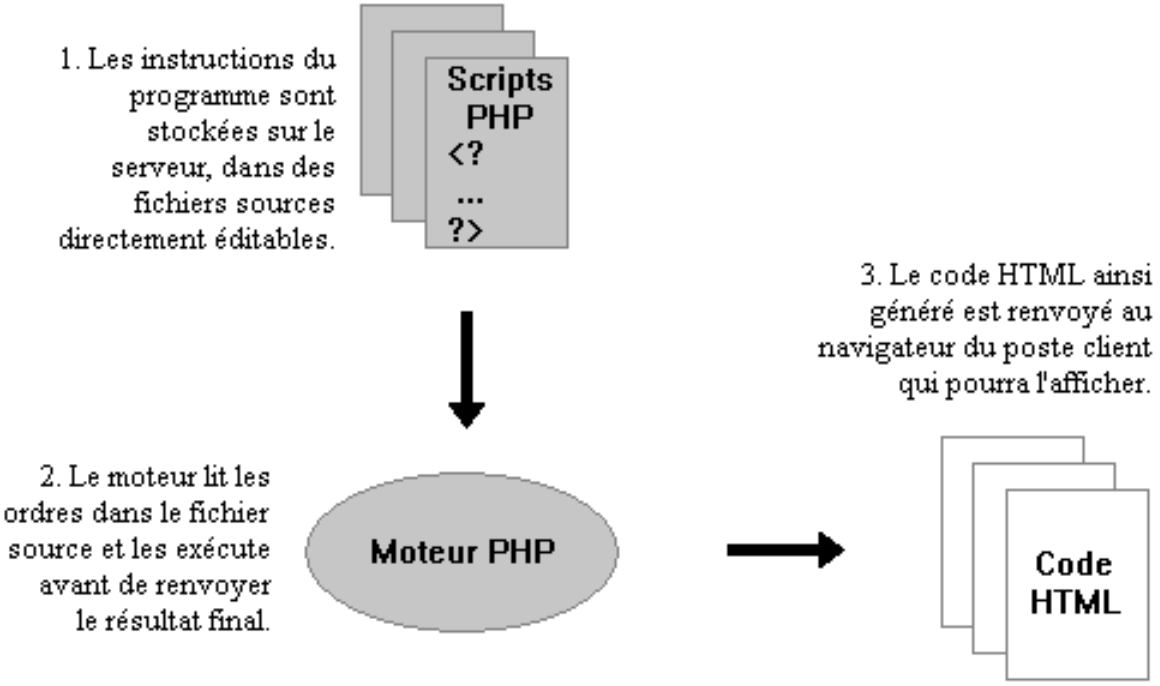
- Créé par Rasmus Lerdorf en 1994 pour des besoins personnels (***P****ersonnal* ***H****ome* ***P****age*) ;

- En 1997, le projet devient un travail d'équipe et l'interpréteur est réécrit par Zeev Suraski et Andi Gutmans pour donner la version PHP3, version qui s'est rapidement imposée et devient PHP (***H****ypertext* ***P****re****P****rocessor*) ;

- La dernière version en date est la PHP4 (2000). Elle intègre en mode natif le moteur Zend (société privée créée par Suraskyi et Gutmans). PHP4 s'avère plus rapide, plus fiable et plus complet. Les scripts sont désormais compilés puis exécutés.

3 . Fonctionnement

➢Un script PHP est un simple fichier texte ASCII contenant des instructions incluses dans du code HTML à l'aide de balises spéciales et stocké sur un serveur disposant d'un interpréteur PHP.



➢Ce fichier script doit avoir une extension reconnue par le serveur (.php3, .php4 ou le plus souvent .php ou tout autre extension défini sur le serveur).

4 . Script PHP

➢Pour que le script soit interprété par le serveur, deux conditions sont nécessaires :

• Le fichier contenant le code doit avoir la bonne extension (et non *.html*)

• Le code php contenu dans le code HTML doit être délimité par les balises `<?php` et `?>`

➢Pour des raisons de conformité avec certaines normes (XML par exemple), plusieurs balises peuvent être utilisées pour délimiter un code PHP :

1. **`<?php et ?>`**
2. `<? et ?>`
3. `<script language="php"> et </script>`
4. `<%php et %>`

5 . Exemple

- Script **hello.php** :

```
<html>
   <head>
        <title>Exemple</title>
   </head>
   <body>
        <?php echo "Hello world"; ?>
   </body>
</html>
```

- Si maintenant on regarde le source de la page Web côté client, on y lit :

```
<html>
 <head>
  <title>Exemple</title>
 </head>
 <body>
   Hello world
 </body>
</html>
```

6 . Syntaxe

- La syntaxe de PHP est directement hérité du langage C et du perl :

- séparateur d'instructions -> `;`

- commentaires :

```
/* ...mes commentaires... */
// ...mes commentaires...,
# ...mes commentaires....
```

7 . Les variables

➢Le langage PHP supporte les types de données suivants :

- Scalaires (entier, flottant, chaînes de caractères)
- Tableaux et tableaux associatifs
- Objets (voir cours « Les classes en PHP »)

➢Tous les noms de variable sont précédés d'un **$**.

➢Les variables n'ont pas besoin d'être déclarées.

8 . Les variables scalaires

- Il n'est pas nécessaire de typer les variables, c'est-à-dire de leur définir un type.
- Il suffit de leur assigner une valeur pour en définir le type :
  - entiers: nombres sans virgule
  - réels: nombres avec une virgule (en réalité un point)
  - chaînes de caractères: ensemble de caractères entre guillemets simples ou doubles

9 . Les tableaux

- Les tableaux stockent des données sous forme de liste.

- Les données contenues dans la liste sont accessibles grâce à une **clé** (ou **index**, indifféremment un **entier** ou une **chaîne de caractères**).

- Contrairement à des langages tels que le C, il est possible de stocker des éléments de types différents dans un même tableau.

- Pour créer un tableau, on peut utiliser :

- la fonction **`array()`** ;
- affecter directement les valeurs au tableau.

10 . Portée des variables

- La portée d'une variable dépend du contexte dans lequel elle est définie.

- On distinguera les variables à portée :

- **Globale** (lorsqu'une variable est déclarée à l'extérieur de toute fonction ou de tout bloc d'instruction, elle est accessible (visible) de partout dans ce code)

- **Locale** (Lorsque une variable est déclarée à l'intérieur d'un bloc d'instructions ou d'une fonction, sa portée est alors locale à ce bloc ou cette fonction)

Remarque : les noms de variable sont sensibles à la casse.

11 . Les variables statiques et dynamiques

- Une variable **statique** est une variable locale qui ne perd pas sa valeur à chaque fois que le bloc est exécuté.

- On utilise, comme en C, l'attribut `static` pour déclarer une telle variable :

```
static $toto ;
```

Remarque : Ce type de variables est très utile pour la création de fonctions récursives.

- Une variable **dynamique** prend la valeur d'une variable et l'utilise comme nom d'une autre variable ;

```
$toto = "Hello" ; # $toto vaut Hello
$$toto = "World" ; # $Hello vaut World
```

12 . Les constantes

- Une constante est une variable dont la valeur est inchangeable lors de l'exécution d'un programme.

- Les constantes sont définies grâce à la fonction **`define()`**, dont la syntaxe est la suivante :

```
define("MA_CONSTANTE", "Bonjour") ;
echo MA_CONSTANTE ; # affiche Bonjour
```

- Le nom d'une constante définie, à l'aide de la fonction **`define()`**, ne doit pas commencer par le caractère **`$`** (de cette façon aucune affectation n'est possible).

- On conseille de toujours utiliser des majuscules pour les noms de constante.

13 . Opérateurs et structures de contrôle

- Les opérateurs et structures de contrôle sont identiques au langage C.
- Seule particularité :

- PHP4 définit une boucle **`foreach`**, comme en Perl, pour réaliser une boucle sur les éléments d'un tableau.

- En PHP3 on peut réaliser l'équivalent avec une boucle **`while`** et les fonction **`list()`** et **`each()`**.

Exemple :

```
foreach ($tableau as $cle => $valeur)
{
  echo "$cle => $valeur, " ;
}
```

14 . Affichage

- PHP fournit 3 fonctions permettant d'envoyer du texte au navigateur :

`echo, print` et **`printf`**

- Ces fonctions ont la particularité de pouvoir insérer dans les données envoyées des variables, pouvant être fonction d'un valeur récupérée par exemple, c'est ce qui rend possible la création de pages dynamiques.

Remarques :

L'insertion de code HTML dans des scripts PHP posent régulièrement des problèmes au programmeur en terme de cohérence, maintenance et portabilité. On cherche donc à séparer l'affichage (HTML) à la partie programmation (PHP), vu que d'autre part ce ne sont le plus souvent pas les mêmes personnes qui créent ces différentes parties (designer/développeur).

Les solutions les plus utilisées sont : encapsulation des fonctions d'affichage dans des classes ou utilisation des *templates (*par exemple *smarty)*.